LE SYSTÈME DE DÉVELOPPEMENT

Pour utiliser un microcontroleur dans un système plusieurs étapes sont nécessaires:

1 Le choix du CPU et la configuration de la carte. Ils sont déterminés par le nombre de ports nécessaires ,les fonctions à réaliser et la vitesse souhaitée.

2 L'écriture du logiciel aboutissant à une liste d'octets qui devront être implantés dans la ROM Programme .

3 Le test de ce logiciel dans des conditions aussi proches que possible de la réalité dans laquelle travaillera le microcontroleur.

4 La gravure définitive de la PROM qui sera implantée sur la carte finale.

Passons en revue ces diverses étapes.

L'étape 1 est essentiellement un problème d'électronique qui ne peut être détaillé que sur un exemple.

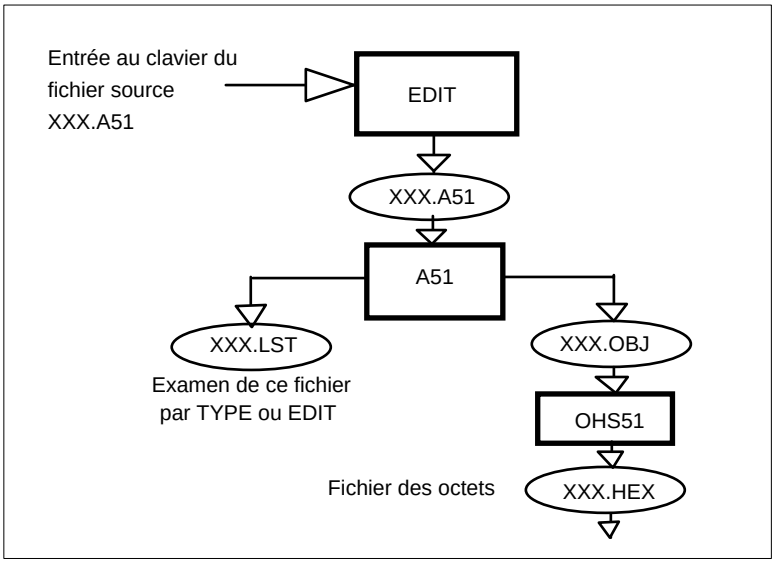
L'écriture du programme : L'ASSEMBLEUR

L'Assembleur est un logiciel implanté dans un ordinateur qui exploite un fichier texte le **programme source** pour créer un **programme objet** en langage machine à partir duquel est enfin construite la liste des octets.

Le programme source écrit en assembleur est un fichier texte qui doit être entré dans la mémoire de l'ordinateur grâce à un **éditeur de texte** quelconque. Il est possible d'utiliser **l'éditeur** du Turbo Pascal , celui **du DOS 6,** ou tout autre logiciel équivalent générant des fichiers ASCII .

De nombreux fournisseurs commercialisent des programmes assembleurs plus ou moins puissants , ASM51 est l'assembleur d'INTEL, le plus souvent cité . On peut trouver sur Internet diverses versions de l'assembleur 8051, tournant souvent sous DOS , sous Windows pour les plus récentes. Certaines sont gratuites par exemple ASM51 de MetalLink ou des versions d'évaluation limitées. Une version d'évaluation limitée à des codes de moins de 4kO est disponible sue le site de Raisonance , cela suffit le plus souvent pour la plus part des applications courantes. Le logiciel proposé ici est A51 (de Keil) de structure très voisine, il est décrit plus loin. On consultera aussi le site d'Atmel .(voir plus loin) .

A partir du fichier source l'assembleur construit 2 fichiers , **xxx.OBJ** en langage machine et **xxx.LST** fichier texte reprenant la liste des instructions du fichier source auxquelles sont ajoutées les numéros des cases mémoire, les codes machine et toutes les informations sur le déroulement de l'assemblage. C'est dans ce fichier en particulier que sont indiquées les erreurs qui peuvent survenir.



A51 est accompagné d'un programme complémentaire qui transforme le fichier OBJ en un fichier xxx.HEX qui contient en clair la liste des octets constituant le programme . La figure ci contre illustre la chronologie des opérations .

Sur Internet :
www.keil.com

www.metaice.com/ASM51/ASM51.htm '(Metal Link)
www.atmel.com/dyn/product
www.raisonance.fr télécharger démo_outils gratuits
Le mot clé 8051 sur Google donne accès à des centaines de sites consacrés à ce circuit.

Utilisation d'un compilateur en langage évolué

L'assembleur est un outil puissant mais spécifique à chaque type de CPU et qui demande un apprentissage assez long. On trouve maintenant dans le commerce ,ou sur le web , des compilateurs C qui convertissent directement un programme écrit en C standard en une suite d'octets pour le 8051. (Voir PACIFIC EXE en freeware) L'emploi de cet outil nécessite cependant une connaissance minimale de la structure du CPU (Le 8051 ne dispose pas d'un écran et clavier et des instructions telles que printf ou scanf doivent être adaptées) ,de plus si l'écriture de programmes complexes faisant appel par exemple à de nombreuses boucles conditionnelles (if then else) ou à des calculs arithmétiques est grandement facilitée, l'assembleur reste l'outil le plus efficace pour la gestion de fils sur les ports. Un programme écrit en C conduit toujours à un code machine plus long que celui qui exécute la même tâche mais écrit en assembleur.

Les test du programme

Un programme écrit directement en assembleur (ou en C puis compilé)ne fonctionne que très rarement du premier coup. Il peut contenir des erreurs non décelables lors de l'assemblage , (par exemple le chargement d'une case mémoire à la place d'une autre) et ne tient pas compte des réactions de l'environnement avec lequel il est relié (temps de réponse des dispositifs par exemple). Le test peut être effectué de diverses façons:

La méthode qui consiste à graver les octets dans une PROM , placer cette PROM sur son socle , lancer le programme , et si cela ne marche pas rechercher les erreurs , puis recommencer , n'est pas viable . Elle est laborieuse , et d'autre part les PROM ne supportent pas très bien (quoiqu'en disent les documentations constructeur) des effacements multiples.

1° Emploi d'un simulateur de programme

C'est un programme annexe qui est capable de simuler le déroulement du programme sous test. L'exécution peut être effectuée complètement ou pas à pas. A chaque étape le contenu de tous les registres internes du CPU est affiché . Il est même possible de simuler les événements extérieurs tels que la détection du signal sur un port . Bien sûr la vitesse d'exécution est très faible par rapport à la vitesse réelle , et il est prudent de modifier la durée des boucles de retard pour réaliser une exécution en un temps raisonnable.

La société RAISONANCE s'est spécialisée dans la diffusion de tels outils. La version mise ici à disposition est une version éducation de SIMIS 51 , d'emploi libre mais limitée à un programme de moins de 1K octets. Un mode d'emploi simplifié est donné dans les pages suivantes.

Un tel simulateur de programme permet de corriger un grand nombre , d'erreurs mais il reste nécessaire de tester le programme dans son environnement réel, c'est le rôle des émulateurs .

On trouvera un produit équivalent sur le site d'Atmel.

2° L'émulateur de PROM

Il n'est utilisable qu'avec un microcontroleur à ROM extérieure (ROMLESS) .

L'EMULATEUR de PROM est une carte comprenant une mémoire RAM de même capacité que la PROM à simuler, accessible de deux cotés. L'un de ces accès est utilisé par le microordinateur via une liaison série ou parallèle, l'autre ne possédant qu'une commande de lecture est associée à un connecteur ayant la configuration du socle de la PROM à simuler. Cet émulateur est parfois appelé carte RAM ROM . Il est commercialisé par de nombreux fournisseurs consulter par exemple lextronic (www.lextronic.fr) , des schémas ont été publiés dans Elektor.

Le connecteur côté ROM est enfiché sur le support de la ROM sur la carte CPU.

La **liaison avec un PC peut être série ou parallèle**. Dans ce dernier cas l'utilisation de la prise d'imprimante est commode et évite l'implantation d'une carte spécifique dans la machine.

Le programme assembleur résident dans l'ordinateur a permis de créer la suite d'octets constituant le programme. Via le câble , et avec l'aide d'un logiciel ad hoc, ces octets sont

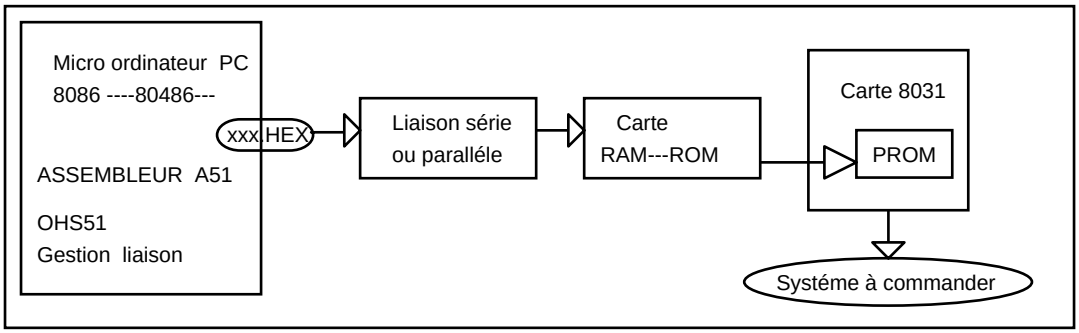
transférés dans la carte RAM/ROM à l'adresse convenable. Le CPU voit la carte RAM/ROM comme une ROM programme et peut donc exécuter les instructions chargées dans cette dernière.

Le système complet est donc formé de:

Un micro ordinateur avec:

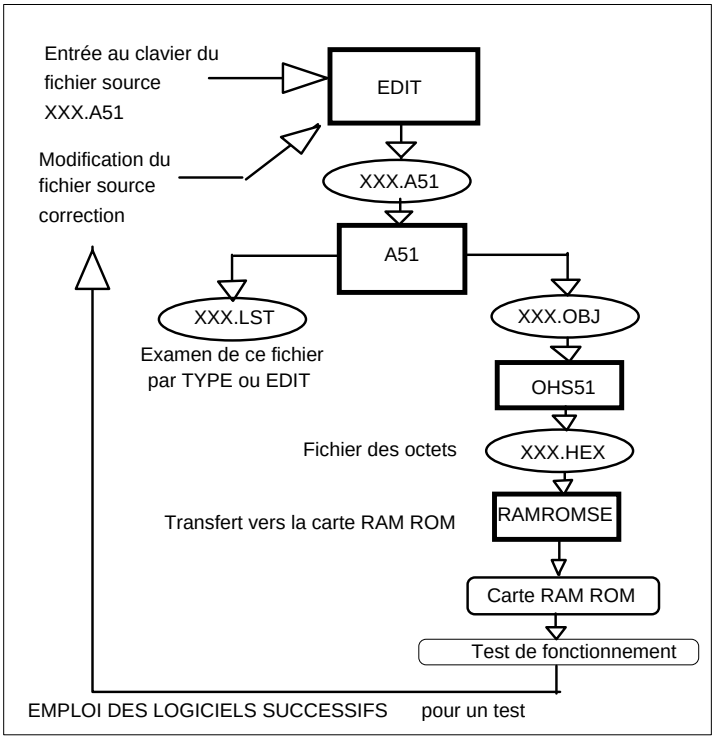
- L'assembleur et outils associés pour créer les octets du programme
- un logiciel de gestion de la carte RAM ROM et un système de liaison PC -carte:
- La carte RAM ROM et sa nappe terminée par le connecteur au brochage de la PROM .

Ce système permet très rapidement de charger un morceau de programme à tester, puis de revenir à l'assembleur pour modification si nécessaire. Le microcontroleur qui tourne est celui de la



SYSTEME de DEVELOPPEMENT

carte finale , la seule différence tient dans le temps d'accès de la carte RAM ROM , mais il est rare que des difficultés surviennent à cause de cela ,du moins pour des CPU ne tournant pas trop vite.



Utilisation des versions Piggy Back

L'utilisation d'un émulateur de PROM n'est possible que si le microcontroleur possède une ROM externe qui malheureusement neutralise au moins 2 ports. Pour tester un programme tournant dans une version ROM ou PROM un émulateur de CPU (paragraphe suivant) est nécessaire (pour les mêmes raisons que plus haut les effacements successifs de la REPROM d'un 8751 ne sont en pratique pas possibles) .Il existe cependant une alternative qui est l'utilisation d'un boîtier PIGGY BACK. Il s'agit d'un microcontroleur classique avec ses bornes normales, qui n'a pas

de ROM interne, mais dont le boîtier possède , en plus des 40 broches, un support de composants de 24 ou 28 broches sur lequel on peut enficher une ROM extérieure.

L'utilisation d'un tel boîtier est très simple, le programme est chargé dans l'émulateur de ROM qui est connecté sur le support supplémentaire. Lorsque la mise au point est terminée une PROM prend la place de l'émulateur .Après un test final le boîtier piggy back surmonté de sa PROM programme est échangé avec un vrai boîtier muni d'une PROM interne (8051 ou 8751).

Le seul inconvénient du système est son coût .Une version Piggy back n'existe que pour un petit nombre de CPU et n'est disponible ,mais coûteuse (300 à 600 F-1999) ,que chez un très petit nombre de fournisseurs en particulier MHS. Il faut noter que bien que coûteux un tel boîtier est indéfiniment réutilisable, et malgré tout bien moins cher qu'un véritable émulateur de CPU.
(ce type de circuit est sans doute maintenant introuvable)

La solution actuelle est de faire appel à des circuits dont la prom interne est une flash. C'est le cas du composant AT89C51 d'Atmel. (voir plus loin)

3° L'émulateur de CPU

C'est le nec plus ultra des systèmes de développement .Il s'agit d'un système complexe piloté par un microprocesseur le plus rapide possible , muni d'un connecteur identique aux broches du CPU émulé , qu'il est capable de simuler complètement à la vitesse réelle. L'émulateur exécute les instructions implantées dans la ROM , mais permet aussi examiner à tout moment le contenu des registres internes grâce à des points d'arrêts programmés.

Il s'agit d'un matériel cher (minimum 1000 Euros) et fragile

Si la vitesse d'exécution n'est pas primordiale , il est possible de réaliser avec un 80451 qui possède de nombreux ports une version lente . En effet les 4 premiers ports peuvent simuler les ports du 8751 , tandis qu'une ROM contenant le programme est pilotée par les 2 ports supplémentaires P4 et P5.. Un tel montage à été décrit récemment dans une revue d'électronique (ELEKTOR).

l'ASSEMBLEUR A51

Cet assembleur est commercialisé par Keil .l Une version d'évaluation est disponible sur www.keil.com

1 La Syntaxe

Chaque ligne a la structure suivante:

[Label:] 8051_Mnémonique [Opérande] [,Opérande][,Opérande]; [Commentaires]

Les éléments entre [ ] sont optionnels .
Le **Label** (étiquette en français) doit commencer par une lettre , il comprend au maximum 31 caractéres , il est suivi de 2 points .Les mots réservés de l'assembleur sont interdits . Un blanc n'est pas ensuite necessaire .

```
Expl        Debut: mov P1,#55H
            sjmp debut
            end  <---------- Un  END  final est nécessaire
            Ce programme charge indéfiniment 55H dans le port 1
```

Les données peuvent être introduites

- en décimal mov A,65
- en Hexadécimal mov A,41H
  un nombre Hexa doit toujours commencer par un chiffre: mov B,0FFH et non FFH
- en binaire mov A,1000001B
- ou même en octal mov A,101o ou 101q
  mais 65 est le code ASCII de la lettre A d'ou une autre notation mov A,'A'

Le logiciel propose un certain nombre de **DIRECTIVES** ,nous ne citerons que les principales :

ORG Pour indiquer à partir de quelle adresse doit être implanté le logiciel on utilise sur la 1ere ligne la commande ORG (Par défaut le programme commence en 0) Par expl :

```
ORG 0          Inutile por un début à zéro.
Debut:mov R1,123
sjmp debut
end
```

La directive ORG peut être utilisée plusieurs fois dans le programme pour effectuer un saut de séquence dans la mémoire.

```
org 0
debut:mov a,#55H
mov P1,#8AH
org 20H
sjmp debut
end
```

fournira après assemblage les codes suivants:

```
Adresse     Codes          Mnémonique
0000         74 55          mov a,#55H
0002         75 90 8A       mov P1,#8AH
.....................................................................
0020         80 DE          sjmp debut
```

END Un END termine obligatoirement le programme. Il y à un seul END ,toutes les instructions situées après un premier END sont ignorées.

EQU Une donnée peut être prédéfinie par EQU:
Nom_de_symbole EQU expression

par exemple Valeur EQU 41H
puis: mov A,valeur
Le contenu de la case d'adresse 41H est chargé dans l'accumulateur ,
mov A,#valeur
chargerait dans A le nombre 41H lui même (adressage immédiat)

SET La directive SET est trés semblable à EQU mais un SET peut faire appel à un SET défini préalablement ce qui est interdit avec EQU:

Valeur SET 45
Valeur SET Valeur/5
Mov A,#Valeur Charge 9 dans A

BIT La directive BIT permet de nommer un bit particulier dans un octet .

alarme BIT CTRL.0 Bit 0 de l'octet CTRL
Voyant BIT 60H le bit est désigné par son adresse absolue (processeur booléen)
lampe BIT 24H.7 adresse absolue et position précisée.

DATA La directive DATA assigne une adresse dans la RAM interne à une donnée dont le nom est précisé.

Nom_de_donnée DATA 4OH
Le contenu de la case 4OH de la RAM interne s'appelle Nom_de_donnée

XDATA Même fonction mais il s'agit d'une adresse externe.

INITIALISATION :

Il est facile d'initialiser des cases mémoire ou les réserver:

DS La directive DS (Defini Space) réserve un espace dans la mémoire
Le symbole $ indique la position courante du pointeur:

Soit :

```
 ORG 0
Debut: MOV P1,#55H
Place: DS 8
sjmp debut
DS 4
Mov A,0
sjmp place+3
End
```

le code machine créé est :

```
debut  0000     75 90 55          MOV P1,#55
place  0003
                        <-------- 8 cases réservées
000B     80 F3                    SJMP DEBUT     (OD+F3=00)
```

```
            000D
                        <-------- 4 cases réservées
            0011    E5 00               MOV A,0
            0013    80 F1               SJMP PLACE+3
```

le pointeur prend la valeur 15+F1=06

DB Cette directive (Defini Byte) permet d'initialiser des cases mémoire-programme . Un label désignant la position est optionnel . Soit à entrer un tableau de 8 valeurs représentant le code ASCII d'un mot :

```
ORG 10
DB 'TRAVAIL'(est équivalent à  DB 'T','R','A','V','A','I','L')
End
```

Correspond au code machine suivant:

```
0010   54 52 41 56 41 49 4C
         T  R  A  V  A  I  L
0017
```

DW (Defini Word) Initialise simultanément 2 cases mémoires dans la mémoire programme Ces 2 cases contiennent un mot de 16 bits .La syntaxe est:

.

[label:] DW expression [,expression][,expression]...

l'expression doit pouvoir constituer des mots; nombre de 4 digits hexa ,chaine de 1 ou 2 caractères.
DW 'Travail' n'est pas admis
DW 7,1245H,'AB' donne la suite d'octets 00 07 12 45 41 42

USING Permet de choisir la "banque" de registres R exploitées par l'instruction suivante utilisant Arx qui est considérée comme une adresse mémoire. Il n'y à pas de changement de banque active ,celle ci est définie par les bits 3 et 4 de PSW .

Exemple:

```
USING 2
MOV R0,A        ; Charge dans A le contenu de R0 de la banque active ,0 par défaut
PUSH AR0        .Sauve dans la pile le contenu du registre R0 de la banque 2 (Adresse 10H)
```

A51 est un logiciel puissant, c'est un macro assembleur permettant de définir et d'utiliser des macro instructions .Pour plus de détails se reporter à la documentation constructeur.

2 Utilisation et exploitation des résultats

L'appel de l'assembleur se fait en tapant A51 suivi du nom du fichier source préalablement créé avec EDIT.

Considérons le fichier suivant qui va nous servir d'exemple, créé avec le nom B:DEFIL

```
ORG 0
mov R0,#0
debut:mov a,R0
mov P1,A
inc R0
mov R1,#3
retr1:mov R2,#255
```

```
 retr2:mov R3,#255
retr3:DJNZ R3,Retr3
Djnz R2,Retr2
Djnz R1,Retr1
sjmp DEBUT
end
```

remarquer que l'on a utilisé indifféremment minuscules et majuscules
Sortir de l'éditeur et appeler:

```
A51 B:DEFIL
```

l'ordinateur travaille puis affiche :

```
MS-DOS MCS-51 MACRO ASSEMBLER A51 V3.94
ASSEMBLY COMPLETE ,NO ERRORS FOUND
C:\A51
```

Si des erreurs se sont produites leur nombre est affiché. Leur nature sera connue en lisant le résultat de l'assemblage Pour cela revenir dans EDIT par EDIT B:DEFIL.LST↵
On obtient:

```
A51 MACRO ASSEMBLER    B:DEFIL

MS-DOC MACRO ASSEMBLER A51 V3.94
OBJECT MODULE PLACED IN B:DEFIL.OBJ
ASSEMBLER INVOKED BY : A51 B:DEFIL


LOC      OBJ         LINE     SOURCE



0000                   1        org 0
0000     7800          2       mov R0,#0
0002     E8            3       debut:mov A,R0
0003     F5 90           4       mov P1,A
0005     O8              5       inc R0
0006     79 03           6       mov R1,#3
0008     7A FF           7       retr1:mov R2,#255
000A     7B FF           8       retr2:mov R3,#255
000C     DB FE           9       retr3:DJNZ R3,Retr3
000E     DA FA          10       Djnz R2,Retr2
0010     D9 F6          11       Djnz R1,Retr1
0012     80EE           12       sjmp DEBUT
                        13      end

A51MACRO ASSEMBLER B:DEFIL


SYMBOL   TABLE     LISTING
-----------     -------      -----------

NAME     TYPE     VALUE        ATTRIBUTES

Debut    C ADDR   0002H         A
P1       D ADDR   0090H         A
RETR     C ADDR   0008H         A
RETR     C ADDR   000AH         A
RETR     C ADDR   000CH         A
```

```
    REGISTER BANK(S) USED :0

    ASSEMBLY COMPLETE NO ERRORS FOUND
    ------------------------------------------------
```

Le fichier B:DEFIL.OBJ créé n'est pas 'listable car ce n'est pas un fichier texte il doit être transformé par OHS51.

Le processus se continue donc en sortant d'EDIT et en tapant:

OHS51 B:DEFIL.OBJ

l'ordinateur travaille et répond :

```
---------------------------------------------------------
A51>OHS51 B:DEFIL.OBJ
MS-DOS MCS-51 OBJECT TO HEX AND SYMBOL FILE CONVERTER V1.1

GENERATING PURE INTEL HEXFILE:B:DEFIL.HEX
C:\>
---------------------------------------------------------
```

Ce fichier est de nouveau un fichier texte lisible par EDIT ou TYPE

```
:100000007800E8F5900879037AFF7BFFDBFEDAFAE7
:04001000D9F680EEAF
:00000001FF
```

Ce fichier présente la suite des octets du programme et sera exploité par le logiciel de transfert .

On remarquera les : qui commencent chaque ligne

- suit un nombre de 2 chiffres representant le nombre de codes sur la ligne,
- un mot de 4 caractéres indiquant à partir de quelle case mémoire se situent ces codes
- 2 chiffres identifiant le type de codes 00 le plus souvent
- les codes à la suite les uns des autres
- un nombre de controle (Checksum) tel que Somme des codes +cheksum=0 modulo 100H

Exemple la premiére ligne précedente :

```
:10    0000        00     7800E8....    ....DAFA     E7
       Adresse-  Type   [ 16 codes Hexa     ]   CheckSum
```

SIMULATEUR DE 8031 SIMICE 51

Ce logiciel permet de simuler le fonctionnement des microcontroleurs de la famille 8051. Les programmes à tester peuvent être exécutés en surveillant à tout moment le contenu des registres internes, c'est donc un outil particulièrement précieux lors de la mise au point de programmes complexes. (Version d'évaluation sur www.raisonance.fr)

Les commandes se présentent sous forme de menus déroulants ,un menu se quitte par la commande **ESC .**

A tout moment on peut consulter une aide en tapant **CTRL ESC**

LANCEMENT DU PROGRAMME et CONFIGURATION

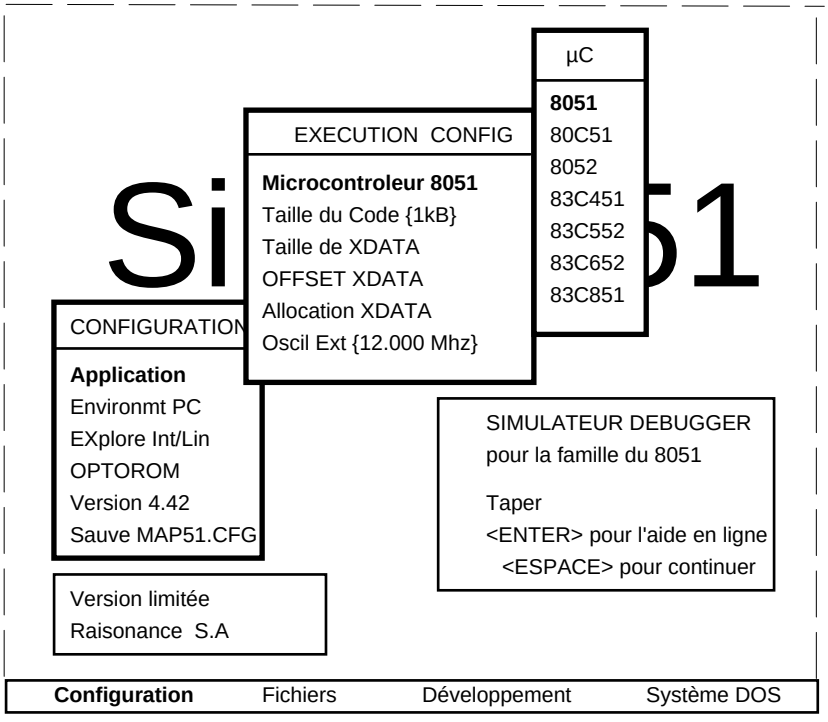
Le programme étant préalablement chargé sur le disque dur il suffit d'appeler **SIMIS :**

Apparaît un écran quadrillé avec en gros caractéres le nom du logiciel SIMIS - 51 et une ligne clignottante indiquant qu'il s'agit d'une version limitée.

On accède à la première page de menu en pressant la barre d'espacement .

Apparaît en bas de l'écran 4 clefs Configuration- Fichiers -Développement -Système DOS dont la premiére est affichée en blanc sur fond noir .

Presser Entrée ↵ , une fenêtre **CONFIGURATION** s'ouvre dans laquelle la ligne **Application** est sélectionnée . La ligne en dessous accessible par la touche ↓ permet de choisir un écran monochrome ou couleurs grace à 3 fenêtres succéssives s'ouvrant en frappant 2 fois ↵ .



Le choix des options étant effectué, on revient au menu initial en fermant les fenêtres par **ESC** . A partir de **CONFIGURATION**

Application , en pressant ↵ on ouvre une seconde fenêtre **EXECUTION / CONFIG** permettant de choisir le microprocesseur ,la taille du code (limitée à 1K pour la version présente du logiciel), la taille de la RAM extérieure, sa position dans le champ mémoire et la fréquence de l'oscillateur. Lorsque la ligne **Microcontroleur** est sélectionnée , la touche ↵ ouvre une troisiéme fenêtre permettant de choisir le CPU (8051,80C51,8052,83451,83552,83652,83851). Il faut noter qu'il n'y a pas de différence ici entre les versions ROM et ROMLESS .

A l'installation du logiciel il est possible de choisir la langue , Français ou Anglais , ainsi que le fournisseur PHILIPS,SIEMENS ou INTEL ce qui donne accès aux différents modéles proposés par ces fabriquants.

Lorsque la configuration compléte est définie (fig ci contre) on ferme les fenêtres par ESC et l'on active par → la clé Fichiers sur la ligne inférieure de l'écran.

CHARGEMENT D'UN FICHIER

En frappant ↵ on ouvre successivement deux fenêtres **ACCES AUX FICHIERS** puis **FORMAT** .

Sélectionner **Charge un programme** dans la première, puis **HEX** dans la seconde ,une nouvelle pression sur ↵ fait apparaître **Nom du fichier ?** sur la ligne inférieure de l'écran

Entrer ce nom avec l'extension HEX, par exemple TESTSMS.HEX Si nécessaire préciser le chemin d'accès. Si le fichier est présent une nouvelle fenêtre s'ouvre au centre de l'écran indiquant que le chargement a bien été effectué .Appuyer 2 fois sur ↵ pour revenir au menu , puis ESC pour effacer la fenêtre ACCES AUX FICHIERS .

Avec la touche → on bascule alors sur la clé **Développement** puis ↵**.**

Nous supposerons à titre d'exemple que le programme suivant à été codé sous forme de fichier .HEX grâce à A51 puis OHS51 ou des logiciels équivalents .

```
MOV P3,#0FFH      FF dans P3
MOV R0,#01                Valeur initiale de R0
BOUCLE:MOV A,R0
MOV P1,A                  Affichage de R0 dans P1
LECTUR:MOV A,P3
CPL A                     Si P3=FF cela donne 0
JZ LECTUR                 Si P3=FF  boucle de lecture de P3
INC R0                    Si P3 n'est pas 0  R0 est incrémenté
MOV R1,#03                Durée de la boucle
DELAI:DJNZ R1,DELAI       Boucle de retard
SJMP BOUCLE       et retour a l'affichage de R0
END.
```

DEVELOPPEMENT

L'écran se présente sous la forme de différentes fenêtres . Une seule d'entre elles est active, son cadre est affiché en surbrillance (Figure suivante) .On passe de l'une à l'autre avec les touches **PAGE↑ et PAGE↓.**

La plus grande est la fenêtre de codes ,c'est elle qui est active au chargement.

FENETRE DE CODE

Elle comprend 5 champs qui sont de gauche à droite:

- L'adresse de l'instruction
- Le code hexadécimal de l'instruction
- une étiquette (Label) éventuelle
- des drapeaux (Flag) de contrôle
- le mnémonique de l'instruction

Le passage d'un champ à l'autre s'effectue avec la touche de tabulation .

Les déplacements dans la fenêtre se font avec les touches flèches ; →, ← , ↓, → ,HOME et END

Les déplacements lointains sont obtenus par:

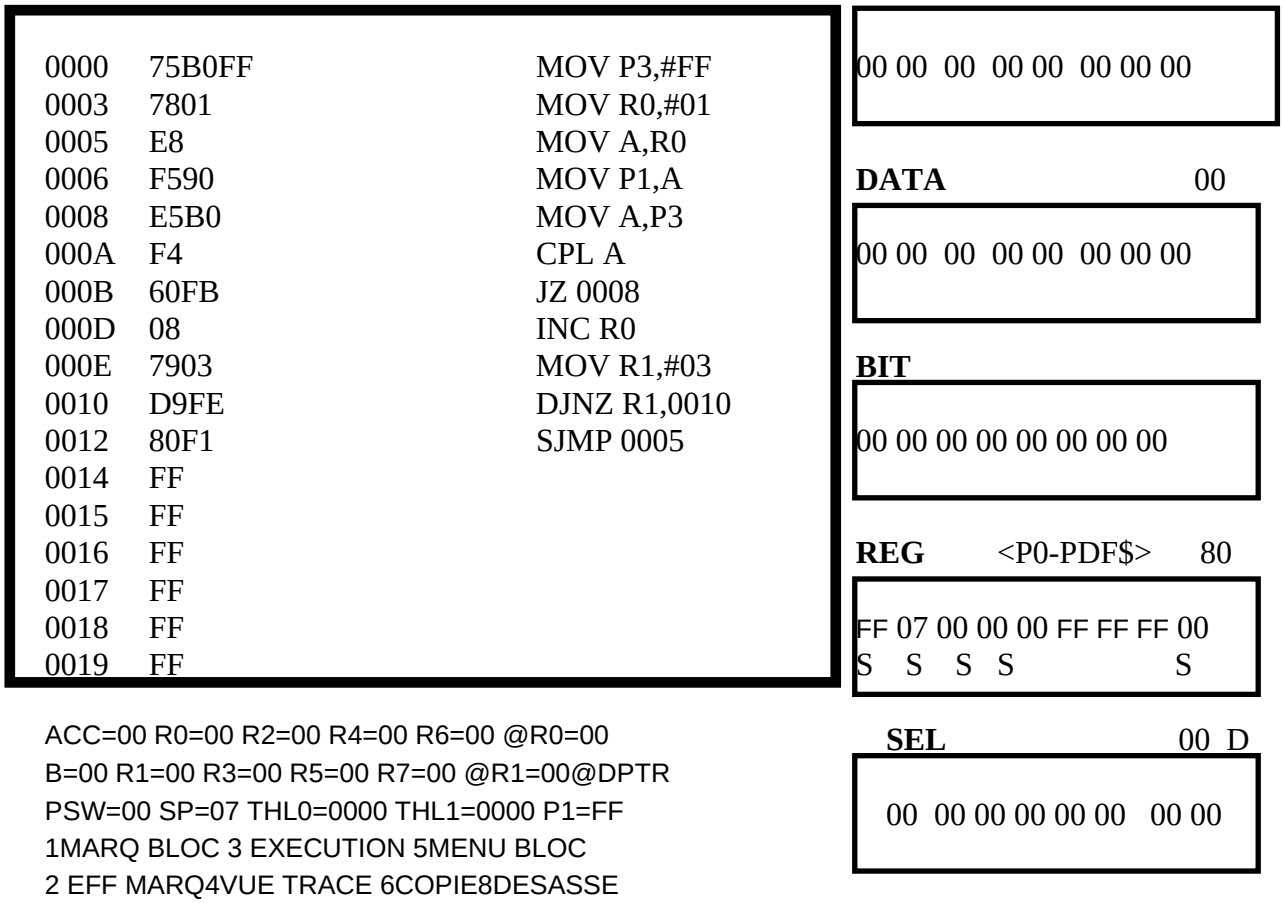
CTRL PG UP pour accéder à la première ligne de code .

CTRL PG DN pour accéder à la dernière ligne du code .

↵ + Numéro de ligne + ↵ pour accéder à une ligne de numéro donné .

Initialement les champs sont chargés à partir du fichier comme on peut le voir sur la figure. Les labels qui éxistaient dans le fichier source ne figurent plus ,en effet le chargement est effectué à partir du seul fichier .HEX qui ne contient que les codes machine.

t>>0 P1 PC=0000 **XDAT** 0000



Les codes peuvent être modifiés :

En se plaçant dans le champ des codes, et en tapant le nouveau code en hexadécimal à la place du précédent.

En se plaçant dans le champ des instructions , et en tapant le mnémonique de la nouvelle instruction . L'assemblage de la ligne est réalisé après avoir pressé la touche ↵. Dans ce cas la longueur de la nouvelle instruction doit être égale ou plus courte que la précédente , sinon le code de l'adresse suivante est perdu .

FENETRE XDAT

Cette fenêtre permet la visualisation et la modification du contenu de la RAM externe (Accessible par les instructions MOVX) . Au départ ce sont les 8 premiéres cases d'adresses 0000 à 0007 qui sont visualisées.

Le déplacement dans cette fenêtre XDAT ainsi que dans les suivantes DATA BIT et REG se fait avec les touches :

→ et ← pour un changement d'adresse d'une unité.
↑ et ↓ pour un changement d'adresse de 8 unités.
↵+ adresse + ↵ pour une adresse quelconque .

La variable active est affichée en surintensité , son adresse est indiquée à droite au dessus de la fenêtre.

FENETRE DATA

Le contenu des 128 premiers octets de la RAM interne sont accessibles dans cette fenêtre.

FENETRE BIT

Elle donne accès aux 256 bits adressables de la RAM interne .

FENETRE REG

Le contenu des 128 octets du SFR est accessible par cette fenêtre.(Adresses de 80 àFF)

FENETRE SEL

Elle permet d'afficher la valeur de 8 variables sélectionnées par l'utilisateur et dont la surveillance est intéressante pour le programme testé.

Le déplacement dans la fenêtre de sélection se fait

- avec les touches → et ←
- avec la touche **Fi** pour sélectionner la variable N°i de la fenêtre.

Le choix de la variable est obtenu en pressant ↵ et le programme demande alors le type de variable : **C** (code) , **X** (donnée de la RAM externe) ,**D** (data de la RAM interne ,**B** (bits adressables), ou **R** (registre du SFR) . Il faut ensuite préciser l'adresse de la variable , dans le SFR (R) les noms tels que P1,DPTR... sont acceptés.

MODIFICATION D'UNE VARIABLE

Il s'effectue en l'activant (affichage en sur intensité) et en tapant la nouvelle valeur . Le contenu des registres principaux est affiché en permanence sous la fenêtre de code .

EXECUTION DU PROGRAMME

L'exécution du programme démarre en appuyant sur **CTRL ESPACE.**

- elle est stoppée en appuyant sur **ESPACE .**
- Il est possible de réaliser une exécution pas à pas en appuyant sur **ALT ESPACE.**

Le menu **exécution** est accessible à partir de la fenêtre de code avec **F3** ,le RESET est notamment obtenu à partir de ce menu.

La modification du compteur programme s'obtient avec le menu **POSITION,** accessible depuis la fenêtre de code avec **F9** .

DRAPEAUX DE HALTE

Il est possible de poser des drapeaux d'arrêt de l'exécution du programme , ils peuvent être posés :

- sur un code :Pour cela se placer dans la fenêtre de code dans le champ des drapeaux de contrôle et taper H . L'exécution s'arrêtera alors juste avant l'exécution de d'instruction considérée.
- Sur une variable: Trois drapeaux sont alors possibles :
  - H halte lorsque le programme accède à cette variable en lecture ou écriture .
  - W : Halte en écriture .
  - R : Halte en lecture .

Remarque: La simulation de l'exécution étant lente, les boucles de retard peuvent durer un temps excessif , modifier en conséquence les compteurs de boucle.

MODIFICATION DES ENTREES DES PORTS PENDANT LA SIMULATION

La modification des entrées des ports permet en particulier de simuler les interruptions externes. Pour cela :

- sélectionner le port considéré (Pi) en appuyant sur **CTRL MAJ Fi**
  le port sélectionné est affiché sur la bande supérieure de l'écran
- Modifier le bit N° j du port sélectionné en frappant **CTRL Fj** .

Exemple :

Reprenons le programme décrit plus haut .Les variables importantes sont le contenu de P1 (affichage) , de P3 (qui reçoit l'action extérieure) , R0 (le compteur) et R1 (le compteur de boucles de retard) .Plaçons les dans la fenêtre SEL ; pour cela cette fenêtre est sélectionnée ,puis on active la premiére variable (mise en surbrillance) et presse la touche ↵ .Une petite fenêtre SEGMENT ? (C,X,D,B,R) s'ouvre.Pour placer P1 on répond R ,(car P1 est un registre du SFR),. puis le nom de la variable P1.

P3 est placé de la même façon en seconde position ,R0 et R1 sont des cases de la RAM interne (DATA) ,il faut préciser leur adresses 00 et 01 car le programme n'accepte pas leurs noms R0 et R1 .

On revient alors dans la fenêtre de codes et pour un premier test on lance le programme en exécution pas à pas après avoir ,par sécurité , effectué un RESET général en pressant F3 puis RESET général .Le tableau suivant présente les contenus successifs de ACC (Accumulateur au début de la ligne en dessous de la fenêtre de codes) , P1 ,P3 ,R0, R1 dans la fenêtre SEL .Le passage d'une ligne à l'autre s'obtient par ALT ESPACE .

Il faut bien noter que le contenu des registres indiqué sur une ligne , est le résultat de l'éxécution des lignes précédentes.

Instruction en surbrillance	ACC	P1	P3	R0	R1	Commentaires
MOV P3,#FF	00	FF	FF	00	00	Etat initial
MOV R0,#01	00	FF	FF	00	00	P3=FF
MOV A,R0	00	FF	FF	01	00	R0 à été chargé
MOV P1,A	01	FF	FF	01	00	ACC reçoit R0
MOV A,P3	01	01	FF	01	00	Affichage de A dans P1
CPL A	FF	01	FF	01	00	ACC à pris la valeur de P3 soit FF
JZ 0008	00	01	FF	01	00	ACC à été complémenté
MOV A,P3	00	01	FF	01	00	A CC étant nul le saut a été effectué
CPL A	FF	01	FF	01	00	
JZ 0008	00	01	FF	01	00	P3 valant toujours FF le saut va s'effectuer de nouveau ,pour l'éviter il faut modifier un bit de P3 ,pour cela on sélectionne le port P3 par CTRL Maj F3 puis CTRL F7 pour mettre à zéro le bit 7 de P3 par exemple. Noter la
	00	01	**7F**	01	00	modification de P3
MOV A,P3	00	01	7F	01	00	Le saut à lieu quand même car la modification de P3 n'a pas été enregistrée dans A
CPL A	7F	01	7F	01	00	
JZ 0008	80	01	7F	01	00	ACC contient le complément de P3
INC R0	80	01	7F	01	00	Le saut n'a pas eu lieu
MOV R1,#03	80	01	7F	02	00	Incrémentation de R0
DJNZ R1,0010	80	01	7F	02	03	Chargement de R1
DJNZ R1,0010	80	01	7F	02	02	Décrémentation de R1
DJNZ R1,0010	80	01	7F	02	01	Décrémentation de R1
SJMP 0005	80	01	7F	02	00	R1 ayant atteint 0 pas de saut
MOV A,R0	80	01	7F	02	00	Saut en début pour affichage de la nouvelle valeur de R0 et examen de P3 etc...

Revenons à l'état initial par RESET (F3) et lançons maintenant une exécution normale par CTRL ESPACE .

En haut de l'écran le compteur d'instructions t= défille rapidement sans qu'une modification de P1 n'apparaisse, car pour le moment P3=FF. Sans arrêter le déroulement, modifions l'un des bits de P3 par CTRL MAJ F3 puis CRTL F1 (modification du bit 1 de F3). On observe alors la fenêtre SEL le défilement de P1 et R0 .Pour arrêter ce défilement il suffit de revenir à l'état initial de P3 par CTRL F1 (il n'est pas nécessaire de faire CTRL MAJ F3 car le port 3 est déja sélectionné, ce qui est indiqué en haut de l'écran .)

QUITTER LE PROGRAMME

Se placer dans le menu principal et taper **S** puis **Q** .

Remarque:

Ce logiciel bien que très pratique n'est pas indispensable, si l'on dispose d'une bonne carte RAM-ROM il est possible de faire tourner le programme par morceaux de plus en plus gros en plaçant de-ci de-là des test de fonctionnement (affichage sur des leds par exemple)° Dès que le programme ne fonctionne plus il suffit de revenir à l'étape antérieure et de suivre pas à pas ce que l'on vient d'ajouter.

BIBLIOGRAPHIE

Les ouvrages traitant du 8051 sont très nombreux je citerai seulement :

Microcontroleurs 8051 et 8052 Bernard Odant DUNOD Tech

Sur Internet www.abcelectronique.com fournit une liste impressionnante de sites consacrés au 8051 ,en particulier de nombreux cours donnés dans les IUT ou universités. Je ne citerai personne pour ne vexer personne , à vous de choisir.

Le 8051 est un microprocesseur remarquable qui malgré son age garde encore tout son intérêt que l'apparition des modèles à flash intégrée confirme encore plus. Mais il est vrai que sa mise en œuvre nécessite un matériel relativement lourd et coûteux , de plus par la taille de son boîtier il conduit à des circuits imprimés assez grands dont la construction nécessite une certaine maîtrise.

Il est certain que la famille PIC de Microchip , pour l'amateur dont l'équipement électronique est limité, est d'accès plus aisé , c'est ce qui a fait son succès.